Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER Computers and Mathematics with Applications 51 (2006) 1103–1112
An International Journal
computers & mathematics
with applications
www.elsevier.com/locate/camwa

A Novel Approach for Bit-Serial AB^2 Multiplication in Finite Fields $GF(2^m)$

JUN-CHEOL JEON, KEE-WON KIM AND KEE-YOUNG YOO*
Dept. of Computer Engineering at Kyungpook National University
Taegu, Korea, 702-701
yook@knu.ac.kr

(Received October 2004; revised and accepted July 2005)

Abstract—This paper presents a new inner product AB^2 multiplication algorithm and effective hardware architecture for exponentiation in finite fields $GF(2^m)$. Exponentiation is more efficiently implemented by applying AB^2 multiplication repeatedly rather than AB multiplication. Thus, efficient AB^2 multiplication algorithms and simple architectures are the key to implementing exponentiation. Accordingly, this paper proposes an efficient inner product multiplication algorithm based on an irreducible all one polynomial (AOP) and simple architecture, which has the same hardware equipment as Fenn's AB multiplier. The proposed bit-serial multiplication algorithm and architecture are highly regular and simpler than those of previous works. © 2006 Elsevier Ltd. All rights reserved.

Keywords—Public-key cryptosystem, Exponentiation, Modular multiplication, Irreducible all one polynomial, Inner products.

1. INTRODUCTION

Finite-field arithmetic operations are currently receiving significant attention due to their important and practical applications in the area of computers and communications, such as error-correcting codes [1], and a number of modern public key cryptographic systems [2]. These cryptosystems include the Diffie-Hellman key exchange and El Gamal encryption schemes, based on modular exponentiations which are implemented by applying modular AB or AB^2 multiplications over finite fields repeatedly as the basic scheme [3–7].

A numbers of studies have presented efficient architectures with algorithms for multiplication based on irreducible AOPs [8–10]. Koc proposed bit-parallel AB multipliers based on a canonical basis [8]. Liu proposed an AB^2 multiplication algorithm using an inner product and parallel two-dimensional cellular architecture [9]. Since Liu's algorithm generates a disordered sequence of coefficients for the resulting polynomial AB^2, it is unsuitable for bit-serial linear feedback shift register (LFSR) architecture. Fenn proposed two types of bit-serial AB multipliers based on

*Author to whom all correspondence should be addressed.
The authors would like to thank the anonymous referees for their valuable suggestions on how to improve the quality of the manuscript. This work was supported by the Brain Korea 21 Project and the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program.

doi:10.1016/j.camwa.2005.07.019

LFSR architecture [10]. These algorithms and architectures have the features of modularity and low complexity but they still need improving with regard to time and space.

Thus, the aim of the current paper is to investigate and develop an effective AB^2 multiplication algorithm and simple and regular architecture for the VLSI implementation of exponentiation in GF(2^m), which is the key operation for public key cryptosystems. A one-dimensional linear feedback shift register (LFSR) is well matched to such criteria, and it has already been applied to many areas [10,11].

In this paper, we propose an efficient multiplication algorithm based on an inner product that computes AB^2 multiplication, plus simple hardware architecture. The important innovation of our work is that the proposed bit-serial multiplier has the same hardware requirements and latency as Fenn's multiplier. The time complexity of a modular exponentiation operation, using the proposed architecture, is 33% lower on average compared to Fenn's architecture.

The remainder of paper is as follows: The mathematical background and Fenn's architecture are described in Section 2. In Section 3, the proposed algorithm and architecture are presented. Section 4 presents a discussion, together with a performance comparison between the proposed LFSR AB^2 multiplication architecture and LFSR AB multiplier presented in [10]. Finally, the conclusion is presented in Section 5.

2. MATHEMATICAL BACKGROUND AND REVIEW OF FENN'S ARCHITECTURE

In this section, we introduce the mathematical background in the finite fields, the AB multiplication algorithm and the hardware architecture proposed by Fenn *et al.* [10].

2.1. Finite Fields

A finite field or Galois field (GF), which is a set of finite elements, is closed based on commutative law, associative law, and distributive law, thereby enabling addition, subtraction, multiplication, and division. Finite fields have received a lot of attention due to their important and practical applications in high bit-rate digital communications. When considered from a hardware implementation point of view, the field GF(2^m) containing 2^m elements is of particular interest for computer applications [12]. Although the addition in GF(2^m) is bit-independent and relatively straightforward, multiplication in GF(2^m) is a more complex and difficult task to carry out efficiently. Therefore, the efficient performance of multiplication in GF(2^m) has attracted considerable research interest.

In order to express the elements in finite fields, there are three major bases; normal basis, dual basis, and polynomial basis. Since a polynomial basis multiplier does not require a basis conversion, it can be readily matched to any input or output system. Also, due to the regularity and simplicity of a polynomial basis multiplier, its design and expansion for high-order finite fields are easier to realize than with dual or normal basis multipliers.

Let $p(x) = x^m + x^{m-1} + \cdots + x + 1$ be an irreducible AOP of degree m over GF(2). $p(x)$ can only be selected as an irreducible AOP for GF(2^m) if and only if $(m+1)$ is prime and 2 is the primitive modulo $(m+1)$. Then, $\{x^{m-1}, x^{m-2}, \ldots, x, 1\}$ forms the polynomial basis for GF(2^m) and any $A \in$ GF(2^m) can be represented by the polynomial basis, such as $A = A_{m-1}x^{m-1} + A_{m-2}x^{m-2} \ldots + A_1 x + A_0$ and $A_i \in$ GF(2) for $i = m-1, \ldots, 2, 1, 0$. One of the AOP properties, $p(x) = x^{m+1} + 1 = 0$, has been suggested in [13] to reduce the complexity of field multiplications. It can be computed simply by $p(x) + xp(x) = (x^m + x^{m-1} + \cdots + x + 1) + x(x^m + x^{m-1} + \cdots + x + 1) = x^{m+1} + 1 = 0$. Let

$$A = A_{m-1}x^{m-1} + \cdots + A_1 x + A_0$$

be an element over GF(2^m). Then, A is also represented as $A = a_m x^m + a_{m-1}x^{m-1} + \cdots + a_1 x + a_0$, where $a_m = 0$, and is denoted by an extended polynomial basis [13].

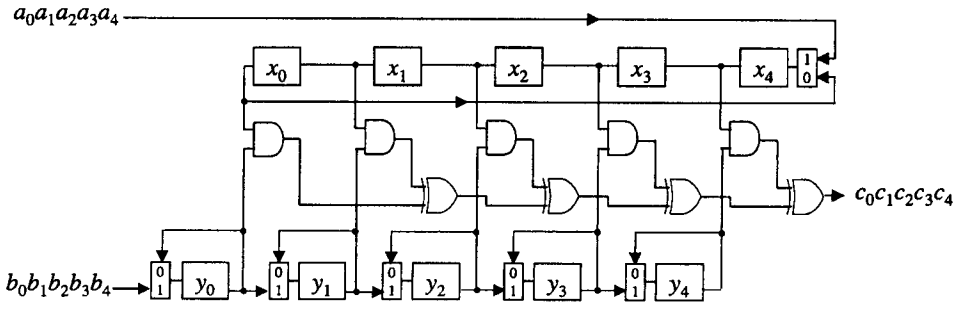

Figure 1. AOPM proposed in [10].

Table 1. The timing table for AOPM in GF(2^4).

Input												Output
Clock	Ctrl Sign.	x_0	x_1	x_2	x_3	x_4	y_0	y_1	y_2	y_3	y_4	
1	1	-	-	-	-	a_4	b_4	-	-	-	-	-
2	1	-	-	-	a_4	a_3	b_3	b_4	-	-	-	-
3	1	-	-	a_4	a_3	a_2	b_2	b_3	b_4	-	-	-
4	1	-	a_4	a_3	a_2	a_1	b_1	b_2	b_3	b_4	-	-
5	1	a_4	a_3	a_2	a_1	a_0	b_0	b_1	b_2	b_3	b_4	c_4
6	0	a_3	a_2	a_1	a_0	a_4	b_0	b_3	b_1	b_4	b_2	c_3
7	0	a_2	a_1	a_0	a_4	a_3	b_0	b_3	b_1	b_4	b_2	c_2
8	0	a_1	a_0	a_4	a_3	a_2	b_0	b_3	b_1	b_4	b_2	c_1
9	0	a_0	a_4	a_3	a_2	a_1	b_0	b_3	b_1	b_4	b_2	c_0

2.2. Review of Bit-Serial AB Multiplication Architecture Proposed by Fenn *et al.*

Let a, b, $c \in \mathrm{GF}(2^m)$, such that $c = ab$ and they represent these elements in the extended polynomial basis as

$$a = \sum_{i=m}^{0} a_i x^i, \qquad b = \sum_{i=m}^{0} b_i x^i, \qquad \text{and} \qquad c = \sum_{i=m}^{0} c_i x^i.$$

Then, the following relationship holds,

$$\begin{bmatrix} c_m \\ c_{m-1} \\ \vdots \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & \cdots & a_m \\ a_m & a_0 & \cdots & a_{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_2 & a_3 & \cdots & a_1 \\ a_1 & a_2 & \cdots & a_0 \end{bmatrix} \begin{bmatrix} b_m \\ b_{m-1} \\ \vdots \\ b_1 \\ b_0 \end{bmatrix}. \tag{1}$$

Consider equation (1) and Figure 1. If the registers in Figure 1 are initialized by $x_{m-i} = a_i$, $y_i = b_i$, for $(i = 0, 1, \ldots, m)$, the first product c_m will immediately be available on the output line. The remaining product bits a_i $(i = m-1, \ldots, 1, 0)$ can be obtained by clocking the upper shift register a further m times.

The AOPM proposed by Fenn *et al.* has been presented by a conventional MSB-first multiplication algorithm [10]. Table 1 shows how the values change in the registers every clock cycle in GF(2^4).

The computation of AOPM over GF(2^m) operates as follows: All x_i and y_i registers are initialized during $(m+1)$ clock cycles. Each operand, a_i, b_i, is inputted through the x_m or y_0

register, respectively. Each value of the x_i and y_i registers are circularly shifted once to the left or right, respectively, during the $(m+1)$ initial clock cycles. Then, the values of the y_i registers are held in the register, while the values of the x_i registers are steadily shifted once to the left during the last m clock cycles. The results are produced from the final input clock cycle to the end of the computational clock cycle. All computations require $(2m+1)$ clock cycles.

3. PROPOSED ALGORITHM AND ARCHITECTURE

In this section, we propose an efficient bit-serial AB^2 multiplication algorithm; thereafter, we present a simple hardware structure based on the LFSR architecture.

3.1. Proposed Inner Product Algorithm

LEMMA 1. *Let $B = b_m x^m + \cdots + b_1 x + b_0$ be an element over GF(2^m). Then, B^2 mod $p(x)$ can be represented by*

$$B^2 \bmod p(x) = \sum_{i=m}^{0} b_{i2^{-1}} x^i, \tag{2}$$

where $p(x) = x^{m+1}+1$. Note that the subscript, k, of b_k, is computed by modulo $(m+1)$ for all computations.

PROOF. The left side of equation (2) is represented by

$$\begin{aligned} B^2 \bmod p(x) &= \left(b_m x^m + b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_2 x^2 + b_1 x + b_0\right)^2 \bmod p(x) \\ &= b_m x^{2m} + b_{m-1}x^{2(m-1)} + \cdots + b_{m/2+1}x^{m+2} \\ &\quad + b_{m/2}x^m + \cdots + b_2 x^4 + b_1 x^2 + b_0 \bmod p(x). \end{aligned} \tag{3}$$

Equation (3) should be reduced by the modulus $p(x)$, i.e., the $m/2$ terms, $x^{2m}, x^{2(m-1)}, \ldots, x^{2(m-(m/2-1))}$ $(= x^{m+2})$ have to be reduced by $p(x)$. Since $x^{2(m-i)} \bmod p(x) = x^{m-2i-1}$, $0 \leqslant i \leqslant (m/2-1)$, this produces

$$\begin{aligned} B^2 \bmod p(x) &= b_m x^{(m-1)} + b_{m-1}x^{(m-3)} + \cdots + b_{m/2+1}x^1 \\ &\quad + b_{m/2}x^m + \cdots + b_2 x^4 + b_1 x^2 + b_0 \bmod p(x). \end{aligned} \tag{4}$$

From equation (4), it should be noted that the first $m/2$ terms have an odd degree of x and the corresponding coefficients of the terms are b_m, $b_{m-1}, b_{m-2}, \ldots$, and $b_{m-(m/2-1)}$ $(= b_{m/2+1})$. Meanwhile, the remaining $m/2+1$ terms have an even degree of x and the corresponding coefficients are $b_{m/2}, b_{m/2-1}, \ldots, b_2, b_1, b_0$.

Meanwhile, the right side of equation (2) is represented by

$$\sum_{i=m}^{0} b_{i2^{-1}} x^i = b_{m\cdot 2^{-1}}x^m + b_{(m-1)\cdot 2^{-1}}x^{m-1} + b_{(m-2)\cdot 2^{-1}}x^{m-2} + \cdots + b_{2\cdot 2^{-1}}x^2 + b_{1\cdot 2^{-1}}x + b_{0\cdot 2^{-1}}. \tag{5}$$

It is shown that each coefficient of term x^i $(0 \leqslant i \leqslant m)$ in equation (4) is equal to that of the corresponding term in equation (5). First, consider the coefficients of the odd degree terms in equation (5). Since m is even, the coefficient of the odd degree term $(m-i)$ is $b_{(m-i)\cdot 2^{-1}}$ when $i = 1, 3, 5, \ldots, m-1$. Then, the subscript $(m-i)2^{-1} = (m-i)/2$ is not an integer. Since $(m+2) \equiv 1 \bmod (m+1)$, this produces $(m-i)/2 \bmod (m+1) = (m(m+2)-i)/2 \bmod (m+1) = (m^2+2m-i)/2 \bmod (m+1) = (1+2m-i)/2 \bmod (m+1) = m-(i-1)/2$. As a result, we have $b_{(m-i)\cdot 2^{-1} \bmod (m+1)} = b_{m-(i-1)/2}$. Thus, the coefficients of term x^{m-i} with an odd degree i are $b_m, b_{m-1}, b_{m-2}, \ldots$, and $b_{m/2+1}$ when $i = 1, 3, 5, \ldots, m-1$.

Now, consider the coefficients of the even degree terms in equation (5). The coefficient of the even degree term $(m-i)$ when $i = 0, 2, 4, \ldots, m$ is $b_{(m-i)\cdot 2^{-1}}$. Since $(m-i)$ is even for an even i, the subscript $(m-i)2^{-1} = (m-i)/2$ is always an integer for an even i. Thus, the coefficients $b_{(m-i)/2}$ of term x^{m-i} with an even degree i when $i = 0, 2, 4, \ldots, m$ are $b_{m/2}, b_{m/2-1}, \ldots, b_2, b_1, b_0$. Therefore, each coefficient of term x^i $(0 \leqslant i \leqslant m)$ in equation (5) is equal to that of the corresponding term in equation (4). Thus, it can be concluded that Lemma 1 holds.

DEFINITION 1. *Let* $A = \sum_{i=m}^{0} a_i x^i$ *be an element over* GF(2^m). $A^{\langle j \rangle}$ *and* $A^{\langle -j \rangle}$ *denote element* A, *which is shifted circularly by* j *bits to the right and left, respectively, i.e.,*

$$A^{\langle j \rangle} = \sum_{i=m}^{0} a_{i+j} x^i \qquad \text{and} \qquad A^{\langle -j \rangle} = \sum_{i=m}^{0} a_{i-j} x^i.$$

DEFINITION 2. *Let* $B^2 = \sum_{i=m}^{0} b_{i2^{-1}} x^i$ *be an element over* GF(2^m). $B^{2\langle j \rangle}$ *and* $B^{2\langle -j \rangle}$ *denote element* B^2, *which is shifted circularly by* j *bits to the right and left, respectively, i.e.,*

$$B^{2\langle j \rangle} = \sum_{i=m}^{0} b_{i2^{-1}+j} x^i \qquad \text{and} \qquad B^{2\langle -j \rangle} = \sum_{i=m}^{0} b_{i2^{-1}-j} x^i.$$

Based on Lemma 1, we have

$$\begin{aligned} A \cdot B^2 &= \sum_{i=m}^{0} a_i x^i \cdot \sum_{i=m}^{0} b_{i2^{-1}} x^i \bmod p(x) \\ &= \sum_{i=m}^{0} a_i b_{i2^{-1}} x^{2i} \bmod p(x) \\ &= \sum_{i=m}^{0} a_{i2^{-1}} b_{i2^{-1}2^{-1}} x^i. \end{aligned}$$

LEMMA 2. *Let* $A = \sum_{i=m}^{0} a_i x^i$ *and* $B^2 = \sum_{i=m}^{0} b_{i2^{-1}} x^i$ *be elements over* GF(2^m). *The inner product of* A *and* B^2 *is* $A \cdot B^2 = \sum_{i=m}^{0} a_{i2^{-1}} b_{i4^{-1}} x^i$. *Obviously,* $A \cdot B^2 = A^{\langle 0 \rangle} \cdot B^{2\langle 0 \rangle}$.

The coefficients of the inner product $A \cdot B^2$ can be computed in order of the decreasing degree of term x^i from the formula of Lemma 2.

DEFINITION 3. *The* j^{th} *inner product* $A^{\langle 2j \rangle} \cdot B^{2\langle -j \rangle}$ *is defined as*

$$A^{\langle 2j \rangle} \cdot B^{2\langle -j \rangle} = \sum_{i=m}^{0} a_{(i+2j)2^{-1}} b_{(i2^{-1}-j)2^{-1}} x^i.$$

THEOREM 1. *Let* $A = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$ *and* $B = b_m x^m + b_{m-1} x^{m-1} + \cdots + b_1 x + b_0$ *be elements over* GF(2^m), *which is produced by an irreducible AOP of degree* m. *Then, the product of* A *and* B^2 *can be represented by the following recurrence equation,*

$$AB^2 = A^{\langle 0 \rangle} \cdot B^{2\langle 0 \rangle} + A^{\langle 2 \rangle} \cdot B^{2\langle -1 \rangle} + \cdots + A^{\langle 2m \rangle} \cdot B^{2\langle -m \rangle}.$$

Therefore, it turns out that

$$AB^2 = \sum_{j=0}^{m} A^{\langle 2j \rangle} \cdot B^{2\langle -j \rangle}.$$

PROOF. The product AB^2 of A and B^2 can be simplified by using the property of $x^{m+1}+1$ and Lemma 1 as follows,

$$\begin{aligned} AB^2 &= \sum_{i=m}^{0} a_i x^i \sum_{i=m}^{0} b_{i2^{-1}} x^i \bmod p(x) \\ &= \left(a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0\right) \\ &\quad \times \left(b_{m/2} x^m + b_m x^{m-1} + \cdots + b_1 x^2 + b_{m/2+1} x + b_0\right) \bmod p(x) \\ &= d_m x^m + d_{m-1} x^{m-1} + \cdots + d_1 x + d_0, \end{aligned} \tag{6}$$

where $d_k = \sum_{i+2j \bmod (m+1)=k}^{a_i b_j}$ and a_i and b_j are the coefficients of A and B^2, respectively.

Based on Definition 3, we have

$$\begin{aligned} A^{\langle 2j\rangle} \cdot B^{2\langle -j\rangle} &= \sum_{i=m}^{0} a_{i+2j}x^i \cdot \sum_{i=m}^{0} b_{i2^{-1}-j}x^i \bmod p(x) \\ &= \sum_{i=m}^{0} a_{(i+2j)2^{-1}} b_{(i2^{-1}-j)2^{-1}} x^i \\ &= \sum_{i=m}^{0} a_{(i+2j)/2} b_{(i/2-j)/2} x^i. \end{aligned}$$

Thus, the right side of Theorem 1, $\sum_{j=0}^{m} A^{\langle 2j\rangle} \cdot B^{2\langle -j\rangle}$ is an element over GF(2^m), which is produced by an irreducible AOP $p(x)$ of degree m. It can be written as follows,

$$\sum_{j=0}^{m} A^{\langle 2j\rangle} \cdot B^{2\langle -j\rangle} = \sum_{j=0}^{m} \sum_{i=m}^{0} a_{(i+2j)/2} b_{(i/2-j)/2} x^i. \tag{7}$$

Let $\sum_{j=0}^{m} A^{\langle 2j\rangle} \cdot B^{2\langle -j\rangle} = c_m x^m + c_{m-1}x^{m-1} + \cdots + c_1 x + c_0$. Then, from equation (7), the coefficient c_k of term x^k $(0 \leqslant k \leqslant m)$ is

$$\begin{aligned} c_k &= \sum_{j=0}^{m} a_{(k+2j)/2} b_{(k/2-j)/2} \\ &= a_{k/2}b_{k/4} + a_{(k+2)/2}b_{(k/2-1)/2} + \cdots + a_{(k+2j)/2}b_{(k/2-j)/2} + \cdots + a_{(k+2m)/2}b_{(k/2-m)/2}. \end{aligned} \tag{8}$$

To prove that $c_k = d_k$ for $0 \leqslant k \leqslant m$, it must be shown that $(s+2t) \bmod (m+1) = k$ for all terms $a_s b_t$ of the coefficient c_k of equation (8), i.e., each term $a_s b_t$ is one among the terms of the coefficient d_k of equation (6). So, for an arbitrary term $a_{(k+2j)/2}b_{(k/2-j)/2}$, $0 \leqslant j \leqslant m$, of the coefficient c_k, we have $(k+2j)/2 + (k/2-j)/2 \bmod (m+1) = k$. Thus, term $a_{(k+2j)/2}b_{(k/2-j)/2}$ is one among $(m+1)$ different terms of the coefficient d_k of equation (6). It is proved that $c_k = d_k$ for $0 \leqslant k \leqslant m$. Therefore, it turns out that $AB^2 = \sum_{j=0}^{m} A^{\langle 2j\rangle} \cdot B^{2\langle -j\rangle}$. The following shows an example of Theorem 1, when $m = 4$.

EXAMPLE 1. Suppose $m = 4$, $A = a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0$, and $B = b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$. It will be shown that $AB^2 = \sum_{j=0}^{m} A^{\langle 2j\rangle} \cdot B^{2\langle -j\rangle}$. As such,

$$AB^2 = \left(a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0\right)\left(b_2x^4 + b_4x^3 + b_1x^2 + b_3x^1 + b_0\right) \bmod x^5 + 1.$$

For example, if c_2 and d_2 are considered as the coefficients of x^2 for $k = 2$. From the above, it is found that $d_2 = a_4b_4 + a_3b_2 + a_2b_0 + a_1b_3 + a_0b_1$. For all terms $a_s b_t$ of d_2, it is satisfied that $(s+2t) \bmod 5 = k = 2$.

In addition, for coefficient c_2, the following results from equation (8) can be shown,

$$\begin{aligned} c_2 &= \sum_{j=0}^{4} a_{(2+2j)/2} b_{(2/2-j)/2} = \sum_{j=0}^{4} a_{(1+j)} b_{(1-j)/2} \\ &= a_1b_{1/2} + a_2b_0 + a_3b_{-1/2} + a_4b_{-1} + a_5b_{-3/2} \\ &= a_1b_3 + a_2b_0 + a_3b_2 + a_4b_4 + a_0b_1. \end{aligned}$$

Thus, it is shown that $c_k = d_k$ for $k = 2$. Plus it can be proved that $c_k = d_k$ for all k by using the same method. Figure 2 shows all coefficients, from c_0 to c_4 for the AB^2 multiplication $(m = 4)$. Note that c_k is computed by adding all terms in the corresponding column in a decreasing order of subscripts. As shown in Figure 2, a_i is circularly shifted two bits to the right while b_i is circularly shifted two bits to the left $(0 \leqslant i \leqslant 4)$.

$A^{\langle 2j \rangle} \cdot B^{2\langle -j \rangle}$	x^4	x^3	x^2	x^1	x^0
$A^{\langle 0 \rangle} \cdot B^{2\langle -0 \rangle}$	a_2b_1	a_4b_2	a_1b_3	a_3b_4	a_0b_0
$A^{\langle 2 \rangle} \cdot B^{2\langle -1 \rangle}$	a_3b_3	a_0b_4	a_2b_0	a_4b_1	a_1b_2
$A^{\langle 4 \rangle} \cdot B^{2\langle -2 \rangle}$	a_4b_0	a_1b_1	a_3b_2	a_0b_3	a_2b_4
$A^{\langle 6 \rangle} \cdot B^{2\langle -3 \rangle}$	a_0b_2	a_2b_3	a_4b_4	a_1b_0	a_3b_1
$A^{\langle 8 \rangle} \cdot B^{2\langle -4 \rangle}$	a_1b_4	a_3b_0	a_0b_1	a_2b_2	a_4b_3
	c_4	c_3	c_2	c_1	c_0

Figure 2. AB^2 multiplication using Theorem 1 in GF (2^4).

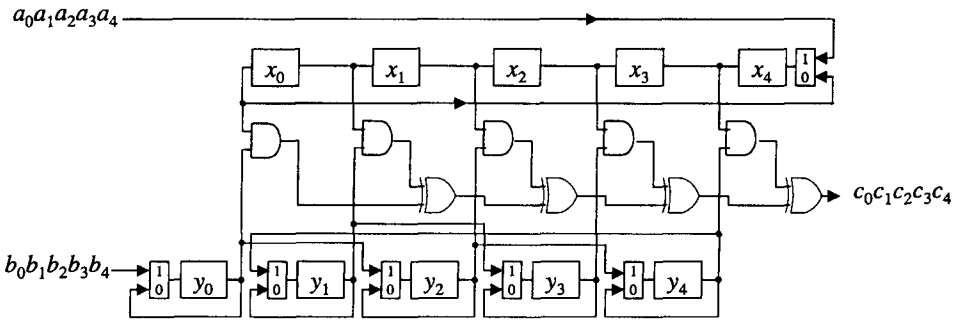

Figure 3. The proposed bit-serial AB^2 inner product multiplier over GF (2^4).

Since the proposed AB^2 algorithm of Theorem 1 does not need further reduction and the rearrangement of the coefficients c_i results in the polynomial $C = AB^2$, the algorithm is better implemented on bit-serial architecture than Liu's algorithm.

3.2. Proposed Bit-Serial AB^2 Multiplication Architecture

This section presents effective bit-serial AB^2 multiplication architecture based on our algorithm which was shown in the previous section. Figure 3 shows the proposed bit-serial AB^2 inner product multiplier.

As seen in Figure 3, the proposed multiplier that computes AB^2 multiplication has nearly the same hardware equipment as that of Fenn's AB multiplier, as shown in Figure 1, except for the wirings. Each y_i $(0 \leqslant i \leqslant 4)$ register and x_0 register have a MUX. The input values, a_i $(0 \leqslant i \leqslant 4)$, are circularly shifted once to the left, while the b_i $(0 \leqslant i \leqslant 4)$ values are circularly shifted 2 bits to the right during $(m+1)$ initialization clock cycles. Whenever the control signal is one and the clock becomes a rising edge, the input values enter each register through a MUX. As soon as all input values enter the registers, the first result, c_4, comes out. Thereafter, the control signal is converted to zero. Then, the input values, a_i $(0 \leqslant i \leqslant 4)$, are circularly shifted once to the left while the b_i $(0 \leqslant i \leqslant 4)$ values are held during the last m computational clock cycles. Our multiplier operates according to the following timing sequence.

In order to compute AB^2, the multiplier carries out the computation for each column in Figure 2 for every clock cycle. The computation over GF (2^m) proceeds as follows. All x_i and y_i registers are initialized during $(m+1)$ clock cycles. Each operand, a_i, b_i, is inputted through an x_m or y_0 register, respectively, based on the relationships between A and B. The values of the x_i registers are circularly shifted once to the left, while the values of the y_i registers are circularly shifted $m/2$ bits to the right, during $(m+1)$ initial clock cycles.

With the last initial clock cycle, the first result is produced. Then, the values of the y_i registers are held, while the values of the x_i registers are steadily shifted once to the left during the last m clock cycles. The results are produced from the final input clock cycle to the end of the

Table 2. The timing table for the proposed inner product multiplier in GF (2^4).

Clock	Input											Output
	Ctrl Sign.	x_0	x_1	x_2	x_3	x_4	y_0	y_1	y_2	y_3	y_4	
1	1	-	-	-	-	a_4	b_4	-	-	-	-	-
2	1	-	-	-	a_4	a_3	b_3	-	b_4	-	-	-
3	1	-	-	a_4	a_3	a_2	b_2	-	b_3	-	b_4	-
4	1	-	a_4	a_3	a_2	a_1	b_1	b_4	b_2	-	b_3	-
5	1	a_4	a_3	a_2	a_1	a_0	b_0	b_3	b_1	b_4	b_2	c_4
6	0	a_3	a_2	a_1	a_0	a_4	b_0	b_3	b_1	b_4	b_2	c_3
7	0	a_2	a_1	a_0	a_4	a_3	b_0	b_3	b_1	b_4	b_2	c_2
8	0	a_1	a_0	a_4	a_3	a_2	b_0	b_3	b_1	b_4	b_2	c_1
9	0	a_0	a_4	a_3	a_2	a_1	b_0	b_3	b_1	b_4	b_2	c_0

computational clock cycle. All computations require $(2m+1)$ clock cycles, which are composed of $(m+1)$ initialization clock cycles and m computational clock cycles.

4. COMPARISON AND DISCUSSION

This section compares the multiplier in [10] and the proposed AB^2 multiplier. The proposed algorithm is basically considered by the concept of the inner product computation, while the algorithm in [10] has been followed by a conventional MSB-first multiplication algorithm. Multiplication is the key operation for implementing circuits for cryptosystems. This is because the process of encrypting and decrypting a message requires modular exponentiation, which can be decomposed into repeated multiplications.

Exponentiation can be computed by using power-sum operations or AB^2 multiplications. A popular algorithm for computing exponentiation is the binary method proposed by Knuth [14]. Let C and A be elements of GF (2^m), the exponentiation of A is then defined as

$$C = A^E, \qquad 0 \leqslant E \leqslant n,$$

where $n = 2^m - 1$. The exponent E is an integer and can be expressed by $E = e_{m-1}2^{m-1} + e_{m-2}2^{m-2} + \cdots + e_1 2^1 + e_0$. There are two ways this can be done. The first is the MSB first exponentiation and the other is the LSB-first exponentiation. Starting from the most significant bit of the exponent, the exponentiation of A can be expressed as

$$A^E = A^{e_0}\left(A^{e_1}\ldots\left(A^{e_{m-2}}\left(A^{e_{m-1}}\right)^2\right)^2\ldots\right)^2.$$

Since equation (9) is actually composed of a series of squaring and multiplication steps, an algorithm for computing the exponentiation is presented as follows.

[MSB-First Exponentiation Algorithm]
Input : A, E, $p(x)$
Output : $C = A^E \bmod p(x)$
Step 1 $C = 1$
Step 2 for $i = m-1$ to 0
Step 3 $C = C^2 \bmod p(x)$
if $(e_i = 1)$ $C = \mathrm{CA} \bmod p(x)$

According to the above algorithm, m squaring always existed, plus the number of modular multiplications of type $C = \mathrm{CA} \bmod p(x)$ is equal to the number of ones in the binary representation of E. This is an integer between 0 and m. Thus, the total number of modular multiplications is at least m and at most $2m$, as stated above. When assuming that a bit pattern of 0 and 1 is

Table 3. Comparison of the number of multiplications between AB architecture and AB^2 architecture for performing modular exponentiation when $m = 1024$ (unit: number of times).

Circuit	AOPM [10]		Fig. 3
The Number of Multiplications	Squaring	Multiplication	1024
	1024	$1 \sim 1024$	
Total Number of Multiplications	$1025 \sim 2048$ 1536 (average)		1024

equivalent, the number of modular multiplications is on average $(m + m/2)$. On the other hand, AB^2 multiplication architecture always operates m times. Therefore, modular exponentiation by an AB^2 multiplier can reduce the number of modular multiplications by at least once and at most m times compared to an AB multiplier. Thus, the key point is how to design simple AB^2 multiplication architecture.

Table 3 shows the number of multiplications involved in performing a modular exponentiation operation. Public key cryptography in GF (2^{1024}) is sufficient in order to achieve a high level of security.

The proposed architecture, which computes AB^2 multiplication, has the same hardware equipment and clock cycle as Fenn's multiplier. Consequently, the exponentiation operation by the proposed multiplier has reduced the number of multiplications, by an average of 33%, as compared to Fenn's multiplier.

5. CONCLUSION

We have proposed an efficient algorithm which carries out AB^2 computation in the finite field GF (2^m). Moreover, we have presented bit serial LFSR architecture for the effective implementation of the inner product algorithm. For implementing the exponentiation, the proposed architecture has shown a superior efficiency to the existing architecture, while at the same time minimizing time complexity by 33%, as compared to the multiplier proposed in [10]. Since our architecture has the features of regularity and modularity, it can be used as an efficient basic algorithm and architecture for division, inversion, and exponentiation which are the core operations in public key cryptosystems.

REFERENCES

1. M. Grangetto, E. Magli and G. Olmo, Robust video transmission over error-prone channels via error correcting arithmetic codes, *IEEE Communications Letters* **7** (12), 596–598, (2003).
2. R.J. McEliece, *Finite Fields for Computer Scientists and Engineering*, Kluwer Academic, New York, (1987).
3. W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Trans. on Info. Theory* **22**, 644–654, (1976).
4. T. El Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. on Info. Theory* **31** (4), 469–472, (1985).
5. A.J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, Boston, MA, (1993).
6. S.W. Wei, VLSI architectures for computing exponentiations, multiplications, multiplicative inverses, and divisions in GF (2^m), *IEEE Trans. Circuit and Systems Analog and Digital Signal Processing* **44** (10), 847–855, (1997).
7. S.W. Wei, A systolic power-sum circuit for GF (2^m), *IEEE Trans. on Comp.* **43** (2), 258–262, (1990).
8. C.K. Koc and B. Sunar, Low complexity bit-parallel canonical and normal basis multipliers for a class of finite fields, *IEEE Trans. Comp.* **47** (3), 353–356, (1998).
9. C.H. Liu, N.F. Huang, and C.Y. Lee, Computation of AB^2 multiplier in GF (2^m) using an efficient low-complexity cellular architecture, *IEICE Trans. Fundamentals E83-A* **12**, 2657–2663, (2000).
10. S.T.J. Fenn, M.G. Parker, M. Benaissa and D. Tayler, Bit-serial multiplication in GF (2^m) using irreducible all-one polynomial, *IEE Proc. Comput. Digit. Tech.* **144** (6), 391–393, (1997).
11. E.R. Berlekamp, Bit-serial Reed-Solomon encoders, *IEEE Trans. IT-2* **6**, 869–874, (1982).
12. R. Lidl and H. Niderreiter, *An Introduction to Finite Field and Their Applications*, CUP, Cambridge, (1986).
13. T. Itoh and S. Tsujii, Structure of parallel multipliers for a class of fields GF (2^m), *Information and Computation* **83**, 21–40, (1989).

14. D.E. Knuth, *The Art of Computer Programming. Volume 2: Evaluation of Powers*, Second Edition, Addisor Wesley, Reading, MA, (1969).